

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

Public Reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE		3. REPORT TYPE AND DATES COVERED Final Report: June 1, 1996 - May 31, 2000	
4. TITLE AND SUBTITLE Systems Software for Irregular Parallel Applications				5. FUNDING NUMBERS DAAH04-96-1-0079	
6. AUTHOR(S) Professor Katherine Yelick					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of California, Berkeley Computer Science Division 777 Soda Hall Berkeley, CA 94720				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211				10. SPONSORING / MONITORING AGENCY REPORT NUMBER ARO Proposal # 35610-MA-YIP 7	
11. SUPPLEMENTARY NOTES The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by the documentation.					
12 a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12 b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The long-term objective of this project was to improve programmability of parallel machines for <i>irregular</i> applications with unpredictable computational costs, pointer-based data structures, dynamically allocated data structures, or asynchronous communications. Example applications include sparse matrix algorithms, adaptive mesh refinement algorithms, and symbolic algorithms. The trend in recent years toward deeper memory hierarchies, including several levels of cache, DRAM, network, and disk, has meant that the more irregular applications have not yet benefited from increasing processor speed as much as more regular applications. Our goal was to provide programmers with high level tools for writing their high performance applications, and our specific tasks addressed three aspects of this problem: application understanding, library development, and compiler optimizations.					
14. SUBJECT TERMS				15. NUMBER OF PAGES 9	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION ON THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL		

20010619 094

MASTER COPY: PLEASE KEEP THIS "MEMORANDUM OF TRANSMITTAL" BLANK FOR REPRODUCTION PURPOSES. WHEN REPORTS ARE GENERATED UNDER THE ARO SPONSORSHIP, FORWARD A COMPLETED COPY OF THIS FORM WITH EACH REPORT SHIPMENT TO THE ARO. THIS WILL ASSURE PROPER IDENTIFICATION. NOT TO BE USED FOR INTERIM PROGRESS REPORTS; SEE PAGE 2 FOR INTERIM PROGRESS REPORT INSTRUCTIONS.

MEMORANDUM OF TRANSMITTAL

U.S. Army Research Office
ATTN: AMSRL-RO-BI (TR)
P.O. Box 12211
Research Triangle Park, NC 27709-2211

- | | |
|--|---|
| <input type="checkbox"/> Reprint (Orig + 2 copies) | <input type="checkbox"/> Technical Report (Orig + 2 copies) |
| <input type="checkbox"/> Manuscript (1 copy) | <input checked="" type="checkbox"/> Final Progress Report (Orig + 2 copies) |
| | <input type="checkbox"/> Related Materials, Abstracts, Theses (1 copy) |

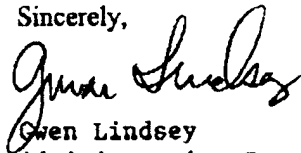
CONTRACT/GRANT NUMBER: DAAH04-96-10079

REPORT TITLE: Systems Software for Irregular Parallel Applications

is forwarded for your information.

SUBMITTED FOR PUBLICATION TO (applicable only if report is manuscript):

Sincerely,



Gwen Lindsey
Administrative Specialist
University of California, Berkeley
Computer Science Division
741 Soda Hall-1776
Berkeley, CA 94720-1776

Enclosure 3

SYSTEMS SOFTWARE FOR IRREGULAR PARALLEL APPLICATIONS

FINAL PROGRESS

KATHERINE YELICK

MAY 24, 2001

U.S. ARMY RESEARCH OFFICE

ARO GRANT #: DAAH04-96-1-0079

UNIVERSITY OF CALIFORNIA BERKELEY

APPROVED FOR PUBLIC RELEASE;

DISTRIBUTION UNLIMITED.

THE VIEWS, OPINIONS, AND/OR FINDINGS CONTAINED IN THE REPORT ARE
THOSE OF THE AUTHOR(S) AND SHOULD NOT BE CONSTRUED AS AN
OFFICIAL DEPARTMENT OF THE ARMY POSITION, POLICY, OR DECISION,
UNLESS SO DESIGNATED BY OTHER DOCUMENTATION

Systems Software for Irregular Parallel Applications

Katherine Yelick
University of California, Berkeley
Computer Science Division
777 Soda Hall
Berkeley, CA 94720
510-642-8900 Phone
510-642-3962 Fax
yelick@cs.berkeley.edu

Research Overview

The long-term objective of this project was to improve programmability of parallel machines for *irregular* applications with unpredictable computational costs, pointer-based data structures, dynamically-allocated data structures, or asynchronous communications. Example applications include sparse matrix algorithms, adaptive mesh refinement algorithms, and symbolic algorithms. The trend in recent years toward deeper memory hierarchies, including several levels of cache, DRAM, network, and disk, has meant that the more irregular applications have not yet benefited from increasing processor speed as much as more regular applications. Our goal was to provide programmers with high level tools for writing their high performance applications, and our specific tasks addressed three aspects of this problem: application understanding, library development, and compiler optimizations.

The first major artifact of this work is the Sparsity system for automatically producing highly tuned sparse matrix algorithms for specific machines and matrices. The second is a set of communication optimizations for the Titanium language, a parallel dialect of Java. This involved both research on new static analyses and better communication subsystems for implementing the parallel constructs on distributed memory multiprocessors and clusters. The funding from this grant targeted lightweight communication substrates, which are described in more detail below along with the Sparsity results. Our group has a history of performing application studies to compliment the basic systems research by suggesting new research problems and aiding with evaluation of systems. As part of this project, we looked at two new irregular applications: the EM algorithm used in the analysis of large data sets and perturbation methods used in modeling dynamical systems. The application studies were very successful in their own right, leading to papers and theses, but we felt that further investment in detailed performance tuning and parallelization would be premature, given that there was still a great deal of innovation possible at the high level algorithm level.

Summary of Most Important Results

Sparsity: Automatic Generation of Sparse Matrix Kernels

Sparse matrix operations dominate the performance of many scientific and engineering applications. In particular, iterative methods are commonly used in algorithms for linear systems, least squares problems, and eigenvalue problems, which involve a sparse matrix-vector product in the inner loop. The performance of sparse matrix algorithms is often disappointing on modern machines because the algorithms have poor temporal and spatial locality and are therefore limited by the speed of main memory. Unfortunately, the performance gap between memory and processing is steadily increasing, as memory performance has not kept pace with the well-known "Moore's Law" speedup that processor performance has enjoyed. Performance is also highly dependent on the nonzero structure of the sparse matrix, the organization of the data and computation, and the exact parameters of the hardware memory system.

We developed a toolkit called Sparsity for the automatic optimization of sparse matrix-vector multiplication. We started with an extensive study of possible memory hierarchy optimizations, in particular reorganizing the matrix and computation around blocks of the matrix. We demonstrated that certain kinds of blocking can be effective for both registers and caches, although the nature of that tiling is quite different due to the differences in size between typical register sets and caches. (Tiling a sparse matrix is entirely different than tiling dense matrices because the matrix is stored as lists of coordinates and values.) Both types of blocking were shown to be highly effective for some matrices, but ineffective on others, and the choice of block size is also shown to be highly dependent on the matrix and machine. Thus, to automatically determine when and how the optimizations should be applied, we employed a combinations of search over a set of possible optimized versions along with newly devised performance models to eliminate or constrain the search to make it practical.

We also considered a common variation of basic sparse matrix-vector multiplication in which a sparse matrix is multiplied by a set of dense vectors. This operation arises, for example, when there are multiple right-hand sides in a linear solver or when a higher level algorithm has been blocked. The introduction of multiple vectors offers enormous optimization opportunities, effectively changing a matrix-vector operation into a matrix-matrix operation. It is well known that for these dense matrices, the latter algorithm has much higher data reuse than the former, and so can achieve much better performance; the same is true in the sparse case.

The Sparsity system was designed as a web service so scientists and engineers could easily obtain highly optimized sparse matrix routines without understanding the specifics of the optimization techniques or how they are selected. It is available from <http://www.cs.Berkeley.edu/cji>.

In addition to the system itself, we did an extensive performance study of over 40 matrices on a variety of machines. The matrices are taken from various scientific and engineering problems, as well as linear programming and data mining. The machines include the Alpha 21164, UltraSPARC I, MIPS R10000, and Power PC 604e. These benchmark results are useful for understanding the performance differences across application domains, the effectiveness of the optimizations, and the costs associated with evaluating our performance models to applying the optimizations. The conclusion is that Sparsity is highly effective, producing routines that are up to 3.1 times faster for a single vector and 6.2 times faster for multiple vectors.

The work on Sparsity resulted in a PhD thesis and two refereed publications during the period of this grant. (Additional papers on this research and follow-on projects have continued to be published after the end of the contract.)

Communication Optimizations for Titanium

Titanium is an explicitly parallel language for parallel scientific computing. It is based on Java and using a Single Process Multiple Data (SPMD) model of computation with a global address space. The global address space allows programmers to build complex distributed data structures that are not possible in message passing languages, and the SPMD model gives an efficient execution strategy without expensive runtime scheduling. Titanium is a dialect of Java, but it is compiled to machine code (though C), rather than to a virtual machine model. It shares the global address space and SPMD parallelism model with Split-C, UPC, AC, and Co-Array Fortran.

The work funded by this grant looked at techniques for efficient implementation of the global address space primitives (mainly remote read and write) on top of modern multiprocessors. In 1996 we performed a study of hardware support for a global address space (as distinct from shared memory, which typically moved data around automatically in cache lines). We examined a spectrum of machine, from those with little or no hardware support for remote memory operations, to those with powerful "communication co-processors," which were sometimes as powerful as the main processor. We found that subtle interactions between the main processor and co-processor could sometimes limit the co-processor, and the more limited hardware was sometimes more predictable and therefore easier to use. In 1998 we looked at theoretical models for network performance, developing a model of the communication performance of store-and-forward networks.

In 1999 we looked at the novel Tera platform as a target for SPMD languages like Titanium. The Tera Multithreaded Architecture (MTA) is a parallel architecture designed to support lightweight threads in a uniform address space. The architecture is in sharp contrast to most other scalable multiprocessors, which have a deep memory hierarchy with caches, local and remote memory, and many cycles of overhead for creating and switching between threads. We explored the question of whether Titanium's computational model could be used effectively on a machine like the MTA, and we described some of the design issues in the MTA implementation of Titanium. Memory latency on the MTA is roughly 100-160 cycles, independent of the address being accessed; this latency is masked by having several threads of execution on each CPU, with automatic switching between threads on each cycle. Titanium applications are designed to map all available parallelism onto a fixed number of coarse-grained processes and to minimize non-local memory references, while the MTA demands a very high degree of parallelism and has little concern over locality. We therefore augmented Titanium's SPMD model by adding loop level parallelization and then considered various mappings of both levels of parallelism onto the MTA, assigning threads to a particular CPU or allowing them to migrate. Ideally, one would like to present all levels of parallelism to the MTA and let it dynamically schedule the threads for good load balance. However, we found that distinguishing between the static, coarse-grained parallelism in Titanium and the dynamic loop level parallelism provides useful information for the thread scheduler, even though locality is not a concern. The reason is that statically-defined parallelism allows for a more equitable distribution of resources, i.e. better load balance. We performed several experiments with our implementation, including both micro-benchmarks and applications studies.

In recent years, market pressures have forced scientists in need of high performance machines to use clusters, whether these are designed as multiprocessors as in the case of the IBM SP line, or built by the end user as a cluster of commodity machines. There has always been an interest in starting high end compute nodes, which in recent years has meant shared memory processors (SMPs). However, these hierarchical machines add yet another level of complexity to the performance optimization space. These CLUMP (Clusters of MultiProcessors) architectures offer improved scalability over clusters of single-processor nodes by potentially replacing network communication latency with far lower bus latency, but the heterogeneous nature of communication on CLUMP systems makes them more difficult to program effectively. We showed that a reimplement of the Titanium runtime library on top of the LAPI (one-sided, lightweight communication layer) on the IBM SP cluster of SMPs achieved performance on a CLUMP that was similar to that of an equivalent cluster of uniprocessors, with no modification to the application code. This gives programmers a single parallel programming model for CLUMPs, rather than a mixture of threads and message passing that is often used. The Titanium compiler generates memory loads/stores when the data is local to the SMP and messages when it is remote. The only overhead of this uniformity is the extra branch in the shared memory case to determine that the data is local, and the storage associated with a pointer to allow it to store the processor number in addition to the memory address of the data. Treating the machine as a flat cluster is attractive, but does not take advantage of the two-level communication hierarchy on a CLUMP. We therefore exposed some alternate models, and believe that ultimately application designers will have to account for the extra level of memory hierarchy to achieve the best performance.

Significant Accomplishments

The following list highlights the accomplishments in this effort.

1. The Sparsity system was released for public use in 2000. The software is available for download from <http://www.cs.berkeley.edu/~ejim/Sparsity>. Sparsity self-tunes implementations of sparse matrix-vector multiplication as well as a sparse matrix times a set of dense vectors. This work is part of a recent trend toward automatic tuning of numerical kernels. A special session of the International Conference on Computational Science was devoted to this topic in May 2001, and although quite recent, it is being transferred to industry. The Matlab system, for example, is starting to use this technology for its dense matrix operation, based on a system called Atlas from U. Tennessee.
2. Titanium has been publicly available for several years. It runs on many platforms, including Uniprocessors, SMPs, and Clusters. The communication and optimizations work produced under this grant has been incorporated into the compiler. Researchers from LBL have used some of the Titanium implementations that are not part of the standard release, namely the Tera MTA and IBM SP. With independent funding from NSF, we are working to make these research prototypes into standard tools. In addition, other language efforts like UPC and Co-Array Fortran may benefit from the work in lightweight communication layers and analysis algorithms.

3. In collaboration with Phillip Collela at NERS/LBNL, we have worked on application level libraries to support for computational fluid dynamics, in particular AMR, through Titanium. That group implemented two fluid solvers based on AMR (one for hyperbolic equations and the other for elliptic equations) and one non-adaptive solver for elliptic equations on infinite domains. The communication optimizations and performance evaluation done as part of this work helped with both the understanding the kinds of algorithms that would perform best on modern machine and improved the quality of these implementations.

Listing of all publications. Please break down publications in the following order:**Papers published in peer-reviewed journals**

A. Krishnamurthy and K. Yelick. "Analysis and Optimizations for Shared Address Space Programs." *Journal of Parallel and Distributed Computation*, 1996.

S. Chakrabarti, J. Demmel, and K. Yelick. "Models and Scheduling Algorithms for Mixed Data and Task Parallel Programs." *Journal of Parallel and Distributed Computing*, Vol. 47, pp. 168-184. 1997.

K. Yelick, L. Semenzato, G. Pike, C. Miyamoto, B. Liblit, A. Krishnamurthy, P. Hilfinger, S. Graham, A. Aiken. "Titanium: A High Performance Java Dialect." *Concurrency: Practice and Experience*, September-November, 1998, pp. 825-36.

E. Deprit and A. Deprit. "Poincaré's methode nouvelle by skew composition." *Celestial Mechanics and Dynamical Astronomy*, vol. 74 (no.3), Kluwer Academic Publishers, 1999. pp. 175-97.

Papers published in non-peer-reviewed journals or in conference proceedings

A. Krishnamurthy, K. E. Schauer, C. J. Scheinman, R. Y. Wang, D. E. Culler, and K. Yelick. "Evaluation of Architectural Support for Global Address-Based Communication in Large-Scale Parallel Machines." *Proceedings of Architecture Support on Programming Languages and Operating Systems*, November, 1996.

K. Yelick, L. Semenzato, G. Pike, C. Miyamoto, B. Liblit, A. Krishnamurthy, P. Hilfinger, S. Graham, P. Colella, A. Aiken. "Titanium: A High-Performance Java Dialect." *ACM 1998 Workshop on Java for High-Performance Network Computing*, February 1998.

R. Wang, A. Krishnamurthy, R. Martin, T. Anderson, and D. Culler. "Towards a Theory of Optimal Communication Pipelines." *SIGMETRICS '98: PERFORMANCE '98 Joint International Conference on Measurement and Modeling of Computer Systems*.

E.-J. Im and K. Yelick. "Model-Based Memory Hierarchy Optimizations for Sparse Matrices." *Workshop on Profile and Feedback-Directed Compilation*, Paris, France, October, 1998.

E. Im and K. A. Yelick. "Optimizing Sparse Matrix Vector Multiplication on SMPs." *SIAM Conference Parallel Processing for Scientific Computing*, San Antonio, TX, March 1999.

Papers presented at meetings, but not published in conference proceedings**Manuscripts submitted, but not published**

A. Deprit, J. Palacian, E. Deprit, and J.-F. San Juan. "The Relegation Algorithm." Submitted to *Physica D*. (To appear.)

Technical reports and theses (not peer-reviewed)

Soumen Chackrabarti. "Efficient Resource Scheduling in Multiprocessors." Phd Thesis, Computer Science Division, University of California, Berkeley, UCB/CSD/ 96/923, November 1996.

E. Deprit. "A Toolbox for Modern Dynamics." PhD Thesis proposal, UC Berkeley, Computer Science Division, October 1996.

E.-J. Im. "Sparsity: A Toolbox for Sparse Matrix Algorithms." PhD Thesis proposal, UC Berkeley, Computer Science Division, April 1998.

A. Krishnamurthy, D. Culler, and K. Yelick. "Empirical Evaluation of Global Memory Support on the CRAY-T3D and CRAY-T3E." UCB//CSD-98-991, 1998.

A. Krishnamurthy. "Compiler Analyses and System Support for Optimizing Shared Address Space Programs." PhD thesis, EECS Department, U.C. Berkeley, May 1999.

Wangeci Ruth Bowman. "Random Projection: A Data Compression Algorithm for EM." Master's Report, Computer Science Division, U.C. Berkeley, December 1999.

Eun-Jin Im. "Optimizing the Performance of Sparse Matrix-Vector Multiplication." PhD thesis, Computer Science Division, U.C. Berkeley, May 2000.

Chang-Sun Lin. "The Performance Limitations of SPMD Programs and Clusters of Shared Memory Multiprocessors." Master Report, May 2000.

List of all scientific personnel showing any advanced degrees earned by them while employed on the project

BOWMAN, WANGECI RUTH
DEPRIT, ETIENNE MAX
IM, EUN-JIN (PhD),now Postdoc
LIN, CHANG-SUN
TANG, KAR MING

Report of Inventions

No inventions during this period.